LAMP-TR-090
CAR-TR-978
CS-TR-4388
UMIACS-TR-2002-68

EAR-99-05844
IIS-00-86116
July 2002

Spatio-Temporal Segmentation of Video by Hierarchical Mean Shift Analysis

Daniel DeMenthon[1] and Rémi Megret[2]

[1]Language and Media Processing (LAMP)
University of Maryland, College Park, MD 20742, USA
daniel@cfar.umd.edu
[2]Laboratoire RFV
INSA Lyon, Villeurbanne, 69621 CEDEX, France
megret@rfv.insa-lyon.fr

Abstract

We describe a simple new technique for spatio-temporal segmentation of video sequences. Each pixel of a 3D space-time video stack is mapped to a 7D feature point whose coordinates include three color components, two motion angle components and two motion position components. The clustering of these feature points provides color segmentation and motion segmentation, as well as a consistent labeling of regions over time which amounts to region tracking. For this task we have adopted a hierarchical clustering method which operates by repeatedly applying mean shift analysis over increasingly large ranges, using at each pass the cluster centers of the previous pass, with weights equal to the counts of the points that contributed to the clusters. This technique has lower complexity for large mean shift radii than ordinary mean shift analysis because it can use binary tree structures more efficiently during range search. In addition, it provides a hierarchical segmentation of the data. Applications include video compression and compact descriptions of video sequences for video indexing and retrieval applications.

The support of NSF grants EAR-99-05844 and IIS-00-86116 is gratefully acknowledged.

1 Introduction and Related Work

One of the goals of video analysis is to find out as much as possible about what is going on in the scene from what was captured by the video. "Finding out what is going on" is more formally called "semantic interpretation". To interpret a scene, one first needs to label independent objects. Boundaries of objects typically correspond to boundaries of color patches in the video; but situations where a foreground object is in front of a background of similar color can also occur. When the camera translates or when objects move independently, boundaries of objects correspond to boundaries across which the optical flow changes in the video, and the patches inside these image boundaries display some consistency of optical flow; but optical flow is notoriously unreliable at object boundaries. To overcome these limitations of motion segmentation and color segmentation and to maximize the chances of correctly extracting objects, researchers have been combining motion and color cues in various ways. The task of dividing video frames into patches that may correspond to objects in the scene is called object-based segmentation [23], layer extraction [22], [21], [34], sprite representation [22] or space-time segmentation [1], [3] [15], [28], [35] (there are arguably subtle differences between these concepts). Color patches produce generalized cylinders in the 3D spatio-temporal pixel volume obtained by piling up frames into a 3D *video stack*. These space-time entities have been called color flows [10], action cylinders [31], and feature trajectories [2] in the literature. We refer to them as *video strands*. Our goal in this paper is to extract these strands and characterize them by color, average radius, and axis position and orientation.

There are three main strategies for space-time segmentation of video sequences:

1. Find spatial regions by segmenting each frame, then track these regions from frame to frame,
2. Track interest points to find their trajectories, then bundle these trajectories,
3. Perform 3D segmentation of the video stack.

The first strategy attempts to discover spatial structures and extend them in the temporal dimension; the second strategy discovers temporal structures and groups them in the spatial dimension; and the third strategy treats the spatial and temporal dimensions equally. The strategies are illustrated in Fig. 1. In this figure, the horizontal dimension represents the sizes of the structures in the spatial dimension, and the vertical dimension represents their sizes in the temporal dimension. The segmentation starts with features that are small both spatially and temporally, near the lower left of the figure. The goal of spatio-temporal segmentation is to group these features into homogeneous structures in the 3D video stack that are large along both the spatial and temporal dimensions, and are therefore at the upper right of the figure. The three strategies correspond to three paths for growing such structures: (1) horizontally, then vertically; (2) vertically, then horizontally; (3) diagonally. Our approach belongs to the third category.

In the first category of spatio-temporal methods, frame-by-frame tracking, there are more variants than we can adequately review here. Typically, the frames are segmented one after the other using motion information. Regions of the previous frame are generally shifted and projected into the current frame by motion compensation, and these projections are compared to the regions of the current frame in various ways to enforce temporal coherence between spatial regions [4], [10], [28], [29], [32], [35]. Another subcategory segments all the frames spatially in a first step, then

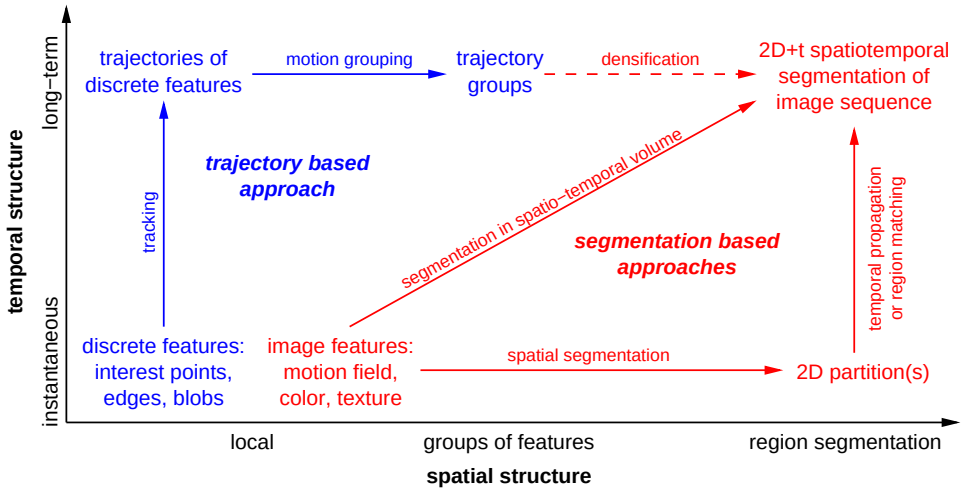


Figure 1: Various paths for growing small elements (pixels, discrete features) into large structures that extend both in the spatial and temporal dimensions of a video stack.

links their regions temporally in a second step [14]. Interesting new variations on these themes, using statistical modeling [22], [23] or subspace constraints [21], have been proposed recently.

In the second category, trajectories of points of interest are extracted; then trajectories with similar long-term motions are bundled together. In [3], the trajectories, called *spatio-temporal flow curves*, are grouped using K-means clustering in a feature space where components represent trajectory curvatures and slope values. In [26], trajectories are grouped by hierarchical agglomerative clustering using a similarity criterion computed from relative positions in several frames, which can accommodate trajectories of different lengths. Subspace factorization has also been applied, e.g. in [13], [18], [6], [17], using a matrix C where each row is a vector of components for the point set of a trajectory. With an affine camera, the trajectories associated with separate rigid bodies lie in separate subspaces, which are obtained by SVD of C or by similar methods. Finally, with a hypothesize-and-test approach, a few trajectories can be randomly grouped to hypothesize a rigid body motion, which is accepted if a large number of trajectories follow the corresponding affine model [11] or a higher-dimensional model [33].

The third category attempts to directly segment the 3D spatio-temporal pixel volume of the video stack. In this way, evidence about spatio-temporal structure can be gathered without favoring one dimension over the other. There is supporting evidence [19] that human vision finds salient structures jointly in space and time. This category is receiving increasing attention, as its computational demands are becoming less of an issue. Such spatio-temporal analysis was pioneered by Adelson and Bergen [1], and Bolles et al. [5]. Recently, Shi and Malik [30], followed by Fowlkes et al.[15], have extended graph cut techniques to the spatio-temporal volume. Graph arcs are considered between all pixels, both within frames and between consecutive frames. The weights of these arcs are decreasing functions of the Euclidean distances between feature vectors of the pixels, with feature components including color components, frame coordinates, and optical flow components. Partitioning into space-time regions is performed by applying K-means clustering in a distance-preserving feature subspace. Finding this subspace involves an eigenvector decomposition of a large system, a problem that requires some simplification, either by considering

only neighboring pixels in the graph [30], or by applying the Nystrom approximation while using only a sampling of the pixels [15].

An alternative strategy in this third category begins by mapping pixels from all the frames at once into a feature space such that pixels corresponding to patches of the same color from the same object that appear in successive frames will be mapped to close neighbors in feature space. Clustering is then performed in the feature space, and pixels are then labeled according to their cluster memberships. In the work presented here, clustering in feature space is produced by an efficient version of mean shift analysis, *hierarchical mean shift*, and the centers of the clusters define what colors and motions should be assigned to the pixels that contributed to the clusters. The result is both a color segmentation and a motion segmentation of the video stack. Greenspan et al. [20] are concurrently developing an approach related to ours in which the clustering is also performed in feature space, but using a Gaussian mixture model initialized by K-means clustering and refined by EM iterations.

In Section 2 we describe our technique for mapping pixels to points of a feature space. In Section 3 we provide an overview of the complete algorithm for clustering the mapped points, and for using the clusters to consistently label the pixels corresponding to moving patches. In Section 4 we introduce hierarchical mean shift analysis, and we present results that support the claim that it produces a significant computational improvement. Section 5 presents segmentation results and outlines how the space-time segmentation can be used to extract concise representations suitable for video retrieval. Finally, Section 6 describes the application of hierarchical mean shift to hierarchical representations of video.

2 Mapping Pixels into Feature Space

Consider a pixel $P_t = (t, x, y)$ at frame t and position (x, y) that belongs to a color patch (Fig. 2). In frame $t+1$, the patch has moved by incremental displacements u in the x direction and v in the y direction, and the pixel P_t of frame t has moved to $P_{t+1} = (t+1, x+u, y+v)$ (u and v can of course be equal to zero). The 3D direction $(1, u, v)$ is the direction of motion of the patch in the video stack. The motion vector $(1, u, v)$ can be found by optical flow computation.

Images of color patches in a video stack produce trajectories along which color components tend to remain stable over a few frames, and the motion vectors $(1, u, v)$ of pixels in these patches tend to remain parallel. Instead of directly using u and v to characterize the motions of color patches, we project the motion vectors on the planes (t, x) and (t, y) of the video stack, and let α_x and α_y be the angles of these projections with respect to the plane (x, y) (see Fig. 2):

$$\alpha_x = 90 - \frac{180}{\pi} \arctan u; \quad \alpha_y = 90 - \frac{180}{\pi} \arctan v \tag{1}$$

When the color patch does not move, both angles are 90^0. Angles approach 0^0 or 180^0 only if the motions are very fast. While the angles obtained from the local optical flow computation characterize *local* orientation trends of trajectories, after the clustering process (described below) the angles of the cluster centers characterize the *global* orientations of the patch trajectories over several frames.

Not only are color and direction of motion approximately constant for a color patch, but in addition the motion vectors are aligned, i.e. the supporting lines of the motion vectors are

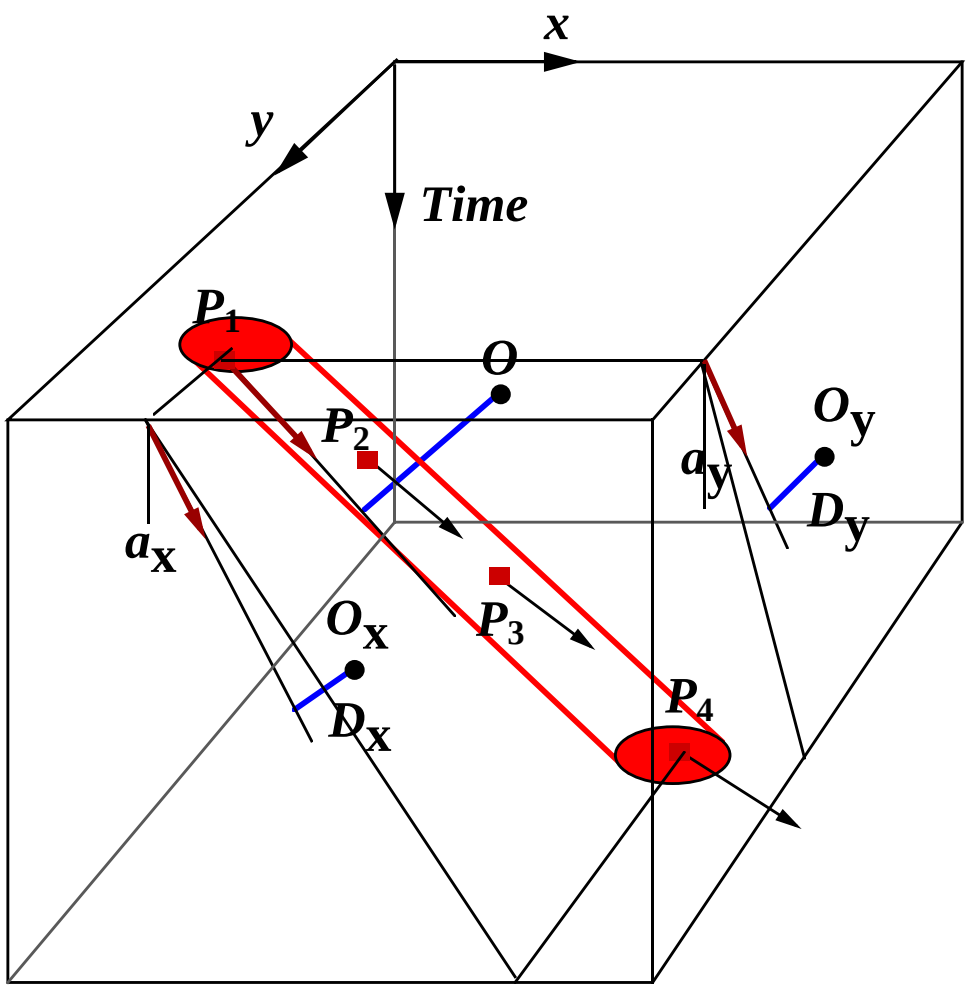


Figure 2: A feature vector with seven components can be defined for each pixel, such that pixels along the trajectory of a color patch in the video stack all have feature vectors that are neighbors in feature space.

approximately superposed (Fig. 2). For each pixel P_i, we project the supporting line of its motion vector on the planes (t, x) and (t, y) and obtain two lines L_x and L_y. Let the point O be at the center of the video stack and let O_x and O_y be the two projections of O onto these planes. We can compute the distance D_x from O_x to L_x and the distance D_y from O_y to L_y for a pixel located at t, x, y in the video stack as follows:

$$D_x = (x - x_{max}/2)\sin\alpha_x - (t - t_{max}/2)\cos\alpha_x \quad (2)$$

$$D_y = (y - y_{max}/2)\sin\alpha_y - (t - t_{max}/2)\cos\alpha_y \quad (3)$$

where t_{max}, x_{max} and y_{max} are the dimensions of the video stack. In the following, D_x and D_y are called the *motion distances* for pixel P. Motion distances have the advantage of defining the positions of pixels in the video stack in a way that is approximately invariant to the motion shifts of the pixels. Using x and y as position features of pixels would provide less compact clusters for pixels of moving patches.

3 Algorithm for Space-time Segmentation

At each pixel of a video stack, seven components are defined: two motion angles, two motion distances, and three color parameters. The CIE L*u*v* color space is used for the color components so that small distances along the color dimensions tend to correspond to perceptually similar colors [7]. We can interpret these quantities as feature components; they define a feature vector which can be represented as a point in feature space. The components are approximate invariants in the color patches; *the points of the feature space that represent pixels of the same color patch moving through time tend to be close together and to form a cluster.* Therefore cluster analysis in this feature space allows us to detect and segment pixels that belong to color patches evolving through time.

Our approach to space-time segmentation is illustrated in Fig. 3: (1) map pixels to points in feature space, (2) determine clusters in feature space (Section 4), (3) assign to each point the label of the cluster to which it belongs, and assign to each pixel of the video stack the label of its mapped point. Since each pixel of a color patch tends to be mapped to the same neighborhood and to belong to the same feature space cluster, color patch pixels tend to be assigned the same label across all the frames of the video stack. Therefore they are *tracked* from frame to frame, in the sense that given color patches in one frame, these patches have labels in that frame, and we can find them in the next frames as the patches with the same labels. See for example Fig. 7, which shows color patches being assigned the same label from frame to frame.

We also find the centers of the clusters in feature space. Since the feature space has seven dimensions, these centers have seven components, which together characterize average values of the motion angles, motion distances, and colors of the patches through time. We obtain a color segmentation of the video stack by replacing the color of each pixel by the color of the cluster to which it is assigned (Fig. 3(c)). Similarly, we can obtain a motion segmentation by assigning to each pixel the motion of its cluster. In addition, we can concisely describe a video clip by its set of cluster centers, whose components describe average characteristics of the video strands.

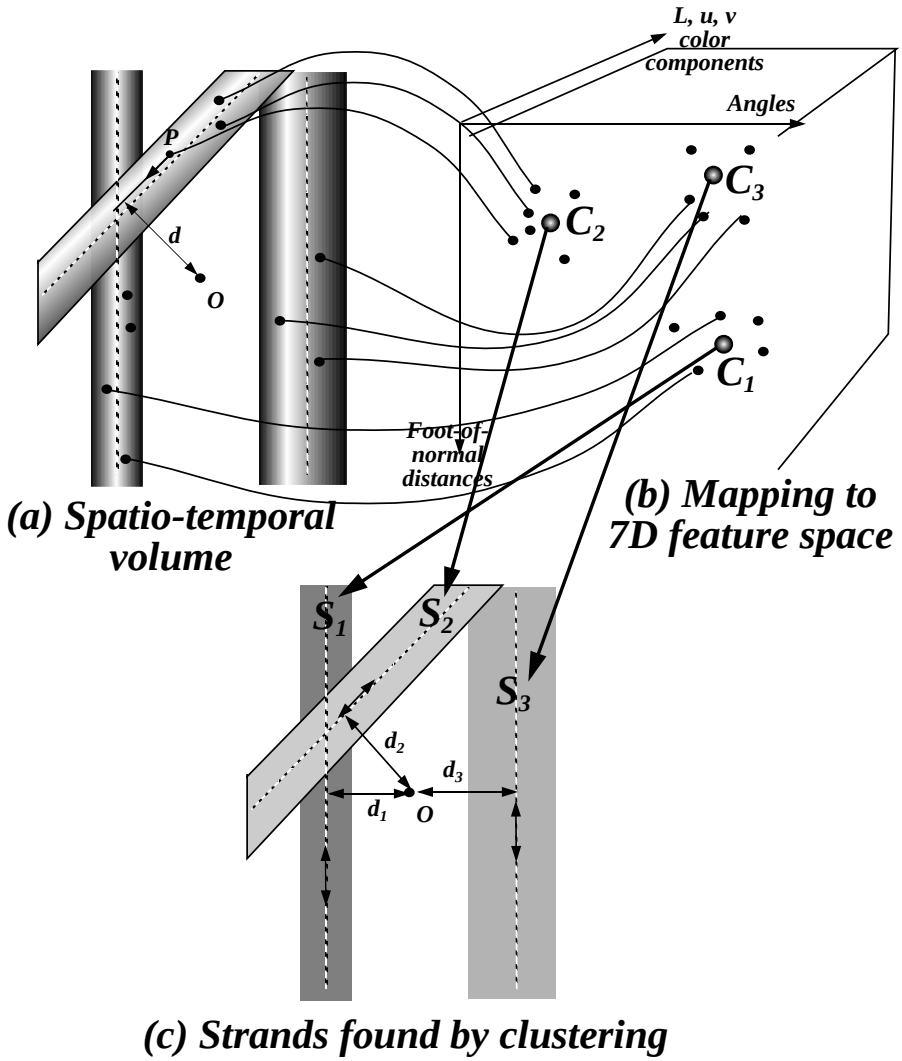


Figure 3: Mapping process between pixels and feature space points, and inverse mapping to obtain segmented regions and video strands.

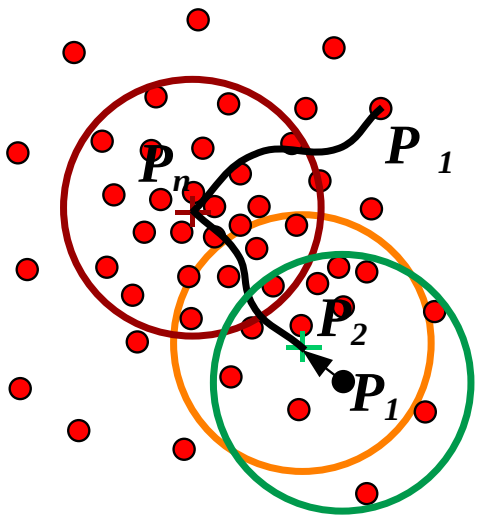


Figure 4: Principle of mean shift analysis: To find the cluster center for point P_1, repeatedly find the centroid of points inside a sphere (initially at P_1) and recenter the sphere on the centroid, until the sphere is stationary. (For Gaussian-kernel mean shift analysis, points further from sphere centers are given exponentially decreasing weights in the centroid calculation.) This is an adaptive gradient ascent in the space of point densities.

4 Clustering by Hierarchical Mean Shift Analysis

Mean shift analysis is a relatively new clustering approach originally advocated by Fukunaga [16], and recently extended and brought to the attention of the image analysis community by Yizong Cheng [12], and then by Comaniciu and Meer [7], [8], [9], who convincingly applied it to image segmentation and frame-by-frame tracking. Refer to these references for details, and to Fig. 4 for the principle of the method. Performing mean shift analysis with a Gaussian kernel is equivalent to performing the following two steps: (1) find a Parzen density estimate of the data set, and (2) find the cluster memberships of individual data points by gradient ascent on the density estimate [24], [9]. In contrast with the classical K-means approach, the clusters that are found are separated by valleys in the point densities, not by artificially defined hyperplanes equidistant between the cluster centers. Finding the natural borders of clusters is important, because such borders in feature space are mapped back to more natural segmentation borders in image space.

Mean shift clustering takes a set of background points and a set of starting points, and requires finding centroids of background points contained in spheres of a given radius R centered on starting points, or centered on centroids found at the previous step. Finding points within spheres requires finding points within distance R of the sphere centers. What is needed is an efficient *range search* algorithm. For this task we use two functions, *nn_prepare* and *range_search*, from the well-written *TSTool* package created by Merkwirth et al. [27]. The auxiliary function *nn_prepare* arranges the background point set into a binary tree structure called an *ATRIA tree*. A set of points is divided into two subsets by the hyperplane halfway between the two farthest points of the set. For each subset, the center C and the enclosing radius r are stored. This is done for each subset until the subset of a branch contains less than a preset number of points. During a range search at the radius R around a point A using the function *range_search*, branches for which the distance

$AC - r$ is larger than R cannot contain points within distance R of A, and are pruned.

However, there is a major obstacle to using a tree structure efficiently in standard mean shift analysis: in order to produce a small number of clusters, mean shift has to be run with a *large radius*, typically up to 1/5 of the span of the largest feature component. For range search that utilizes a tree structure such as the binary ATRIA tree just described or a K-d tree, the cost for N points is $O(N \log N)$ *only for small radii*; for large radii it is closer to $O(N^2)$, because most of the branches of the tree must then be explored. We have adopted a hierarchical mean shift approach to circumvent this problem:

- We first run standard mean shift to completion with a very small radius, starting from all points of the data set and shifting the spheres over the static background of points to reach cluster centers that are local maxima of point densities. In the centroid computations used to compute the shifts, each point is assigned a weight equal to 1. Spheres from several starting points typically converge to the same cluster center, and these points are considered to be members of the corresponding cluster.

- We assign *weights* to these cluster centers, equal to the sums of the weights of the member points.

- We consider the set of cluster centers as a new cloud of points, and recompute a new binary tree. We run mean shift using range search with a larger radius that is a small multiple of the previous radius (we have used a multiplying factor of 1.25 or 1.5). In the centroid computations, the weight of each point is used.

- We repeat the previous two steps until the desired radius size (or the desired number of large regions) is reached.

It turns out that essentially the same method was discovered by Leung et al. in their *clustering by scale-space filtering* approach (see [24, p. 1400, Eq. 22]).

Qualitatively, the segmentation obtained by this technique looks as good as or better than that obtained by standard mean shift. Equally important, a significant speedup is achieved with this method. The reason is that the initial tree handles a very large number N of points, but allows efficient range search because the radius of the range search is small. At subsequent passes, the points are clusters from the previous passes, and their number N' gets smaller at every pass as the radius gets larger; therefore the new tree structure generated for the range search contains N' points, with N' much smaller than N when the radius is large. The complexity of the range search then deteriorates toward $O(N'^2)$, but this is not costly because N' is already quite small when this occurs.

The poor performance of standard mean shift with large radii and the effectiveness of the hierarchical mean shift solution are illustrated in Fig. 5. In this figure, computation time is plotted as a function of the mean shift radius, expressed as a fraction of the data hypercube size, for standard mean shift (diamond plot) and for hierarchical mean shift (triangle plot), for the space-time segmentation of 12 frames of a video, corresponding to 45,000 pixels. The computing time is almost independent of the mean shift radius for hierarchical mean shift, whereas it grows at a rate faster than polynomial for standard mean shift.

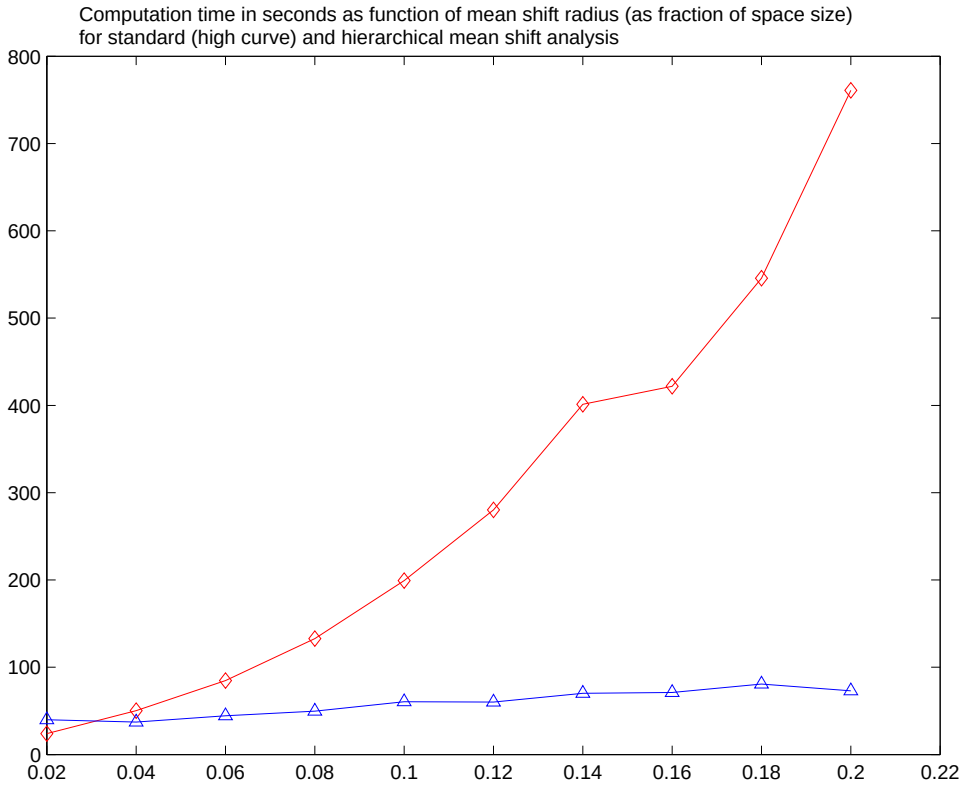


Figure 5: Computation time in seconds as a function of mean shift radius for standard mean shift (higher curve, diamond plot) and for hierarchical mean shift (lower curve, triangle plot). The radius is expressed as a fraction of the feature space size, and is increased to a maximum of 0.2.

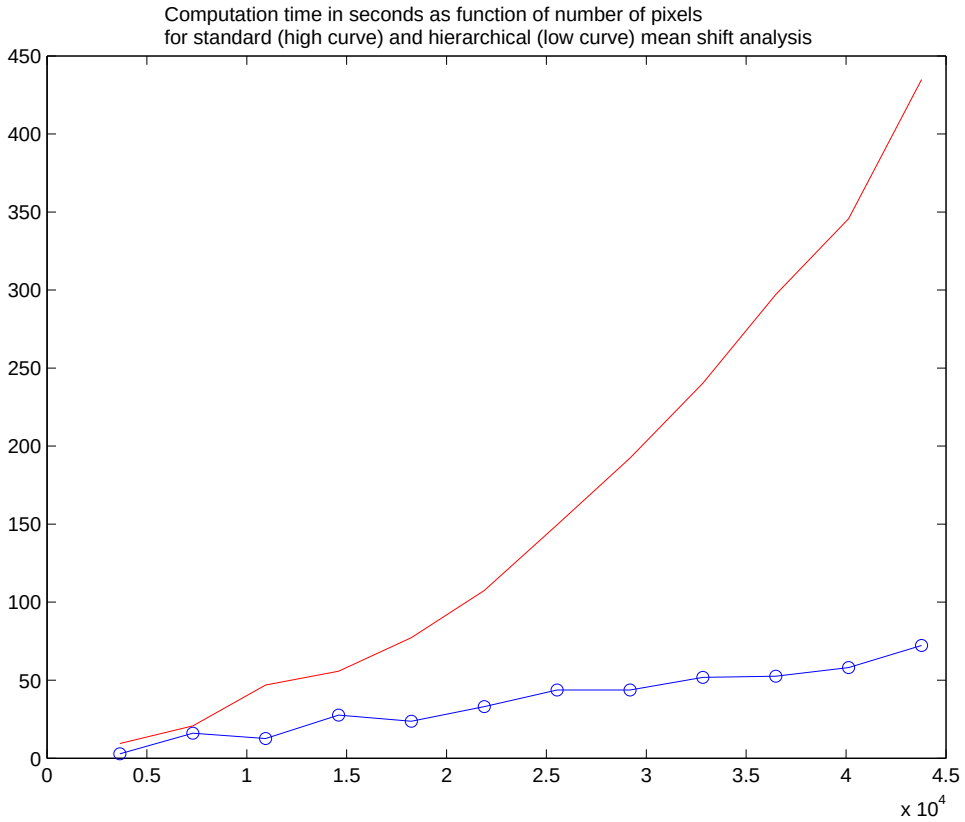
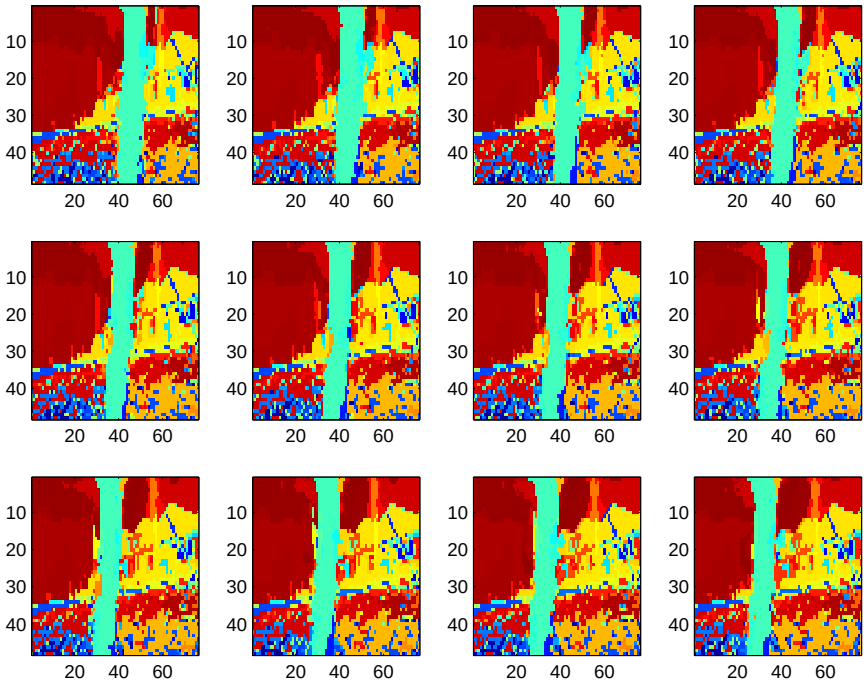
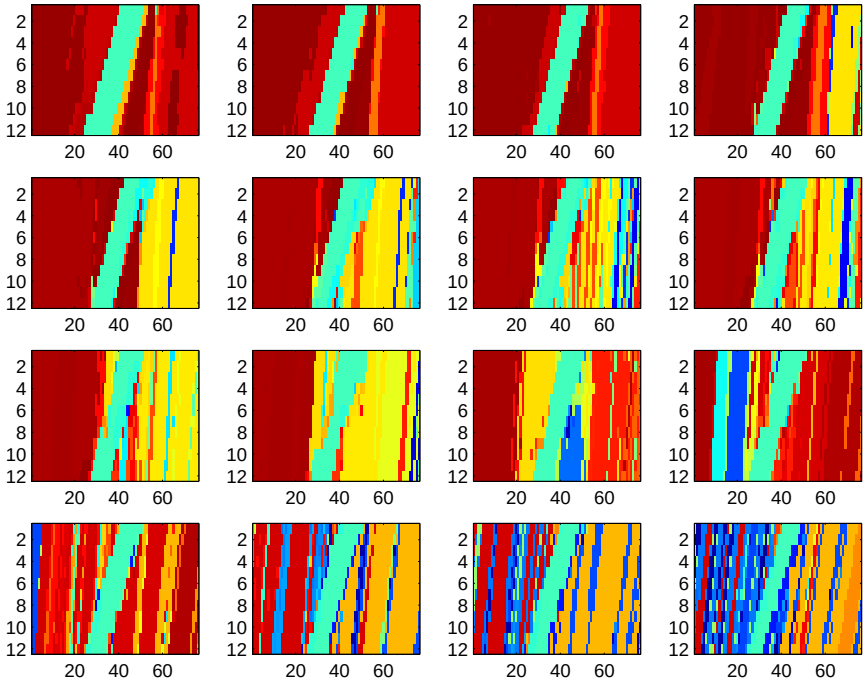


Figure 6: Computation time in seconds as a function of number of pixels for standard mean shift (higher curve, square plot) and for hierarchical mean shift (lower curve, round plot).



(a)



(b)

Figure 7: Consistent labeling of regions, shown with false colors in 12 consecutive frames (a), and shown in 12 cross-sections of the video stack along the row and time dimensions (b). Note that most of the pixels of the tree trunk have been given a single label.

Next, we keep the mean shift radius (i.e., the final radius in the hierarchical mean shift algorithm) constant and equal to 1/6 of the data hypercube size. We repeated our space-time segmentation experiments for a video stack containing an increasing number of frames, from 1 to 12, with the corresponding number of pixels increasing from 4,000 to 45,000. Results for standard mean shift (square plot) and for hierarchical mean shift (round plot) are plotted in Fig. 6. For the largest number of pixels, standard mean shift is almost 10 times slower than hierarchical mean shift. An estimate of the slopes of the log-log versions of the two curves shows that for standard mean shift the computation time increases as the 1.8 power of the number of pixels, i.e. almost quadratically, while for hierarchical mean shift, computation time increases as the 1.1 power of the number of pixels, i.e. almost linearly. The computing time *per frame* for hierarchical mean shift remains *roughly constant* when the number of frames is increased (around 6 seconds per frame in the experiments shown in Fig. 6), whereas standard mean shift becomes impractically slow.

5 Results

We provide examples of video segmentations for the Flower Garden sequence. A list of 7D feature vectors was generated, one per pixel of a video stack of twelve 88×60 frames, with two motion angles computed from optical flow over 20 frames by the Lucas-Kanade method [25], two motion distances computed by the method of Section 2, and three L*u*v color components. The motion components were offset and scaled to range 0-0.8, and distance components to range 0-0.6, while the color components were scaled to range 0-1: instead of using ellipsoids in mean shift, we always use spheres in order to take advantage of the range search machinery, but squeeze the data world along the dimensions that are less important. The importance of motion is decreased because it is less reliable than color, and the importance of distance is decreased even more to avoid subdividing spatially elongated regions. The final radius in the hierarchical mean shift analysis was 1/6, the initial radius was 1/60, and the radius multiplying factor was 1.5 at each pass. Furthermore, we ran the segmentation twice, in order to use in the second pass the cleaner segmented optical flow obtained by the first pass. This led to qualitative improvements in the segmentation results. The code was written in Matlab, with bottlenecks rewritten as C functions with Matlab interfaces. The TSTool functions *nn_prepare* and *range_search* used for computing range search [27] were also written in C with Matlab interfaces. The total processing time per frame was around 15 seconds.

Fig. 7 shows space-time volumes labeled by this algorithm in spatial and temporal cross-sections of the video stack in false color. These volumes are obtained by giving a unique label to each cluster obtained by hierarchical mean shift, and assigning that label to the pixels that were mapped to that cluster. Note that the tree trunk is consistently labeled across the stack, and that the house regions have the same label on the left and right of the tree; this segmentation can accommodate both spatial and temporal occlusion to a certain extent, because of a "Hough transform effect": regions that are disjoint in pixel space can be neighbors in feature space.

Figure 8 demonstrates motion segmentation. Each cluster center has seven dimensions, including two motion angle components. In the figure, the motion angle of each cluster center in the x direction was assigned to all the pixels that contributed to that cluster. Faster lateral motion is coded with a lighter color. Because the camera is translating, the images are in fact depth maps of the scene. Note that the patch of sky in the top left corner of the frames was coded as closer than the rest of the background because texture was generated by hanging branches of the

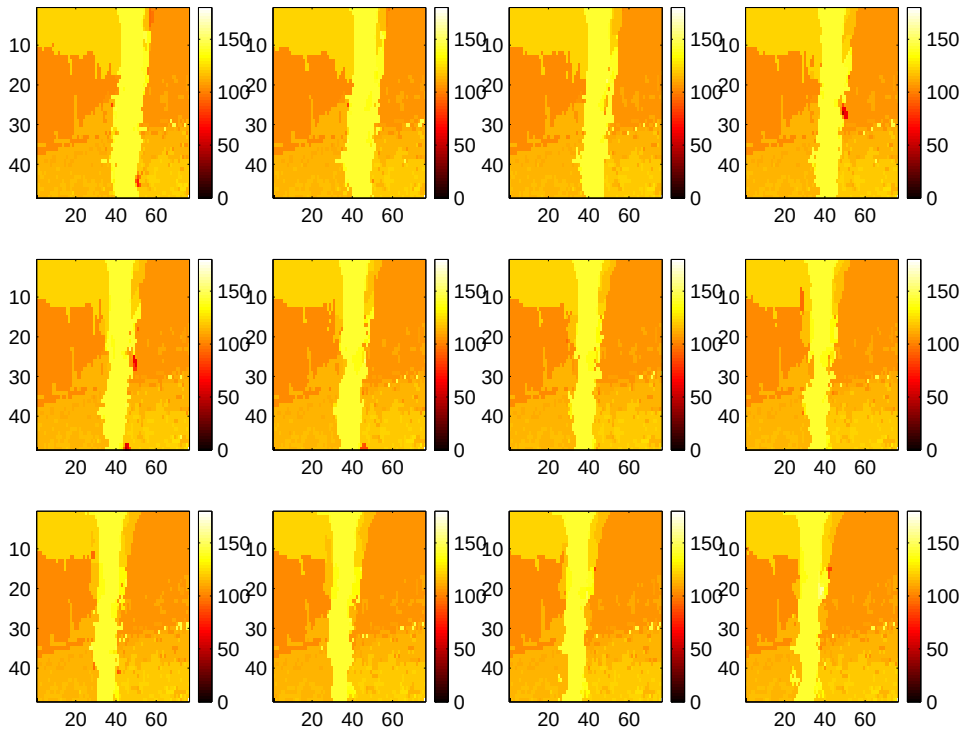


Figure 8: Motion segmentation of video. The color coding shows values of the angles of patch trajectories in the x direction from 0 to 180. These angles increase as distance decreases because the camera translates in the x direction, so lighter colors correspond to closer ranges.

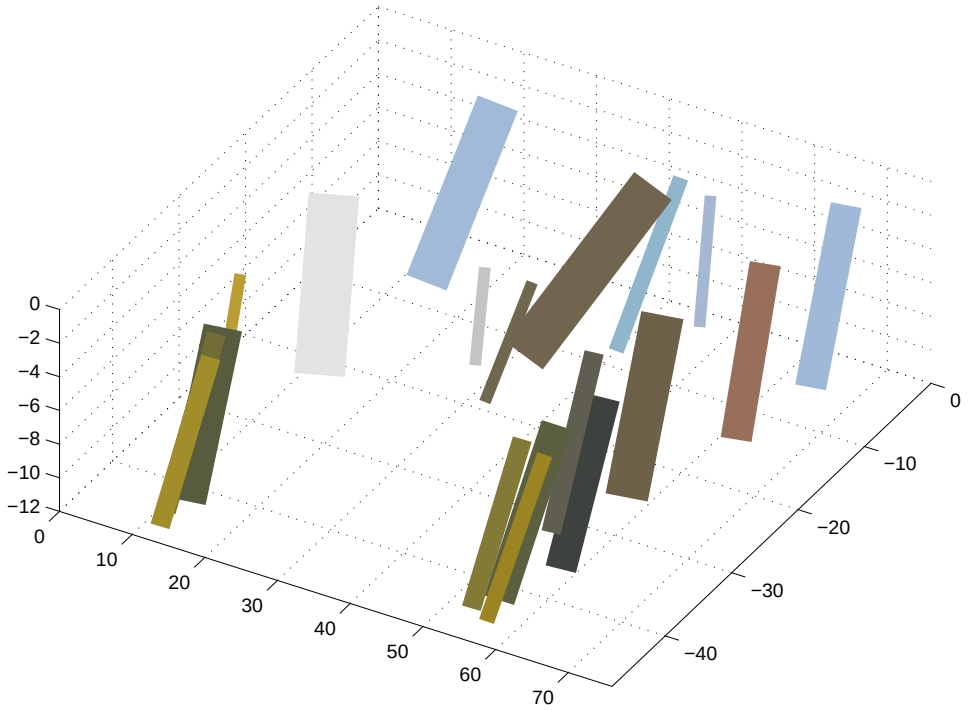


Figure 9: Representation of the Flower Garden sequence as a set of lines with specific thicknesses (video strands). The vertical dimension is time. The thick brown line with the highest slope corresponds to the tree trunk. The blue line on the top left corresponds to a patch of blue sky. It is tilted because it contains tree branches that seem to move because of camera translation. Lines further in the background are almost vertical.

tree. Color segmentation (not shown here) is similarly obtained by assigning color components of cluster centers to their contributing pixels.

The seven dimensions of each cluster center in feature space describe the average color, position and orientation of a color region. These components describe the geometry and color of a straight line in the video stack. The set of lines corresponding to regions that have an average of more than 20 pixels in the frames that they occupy is shown in Fig. 9. This is what we call a video braid representation. This representation can describe one MB of a half-second video with 200 bytes, which can be used as an indexing description. We are investigating retrieval techniques using this description.

6 Hierarchical Segmentation from Hierarchical Mean Shift

Hierarchical mean shift produces a hierarchical segmentation that can be represented as a tree structure. At each pass of the procedure, clusters are merged into new clusters. Each cluster represents a region of the video block, and regions corresponding to new clusters are groupings of regions corresponding to clusters of the previous pass. A fine-to-coarse evolution of the segmentation occurs from pass to pass. With a large enough radius, we obtain a single region corresponding to the whole set of pixels. We demonstrate this evolution on a sequence of twelve frames from the

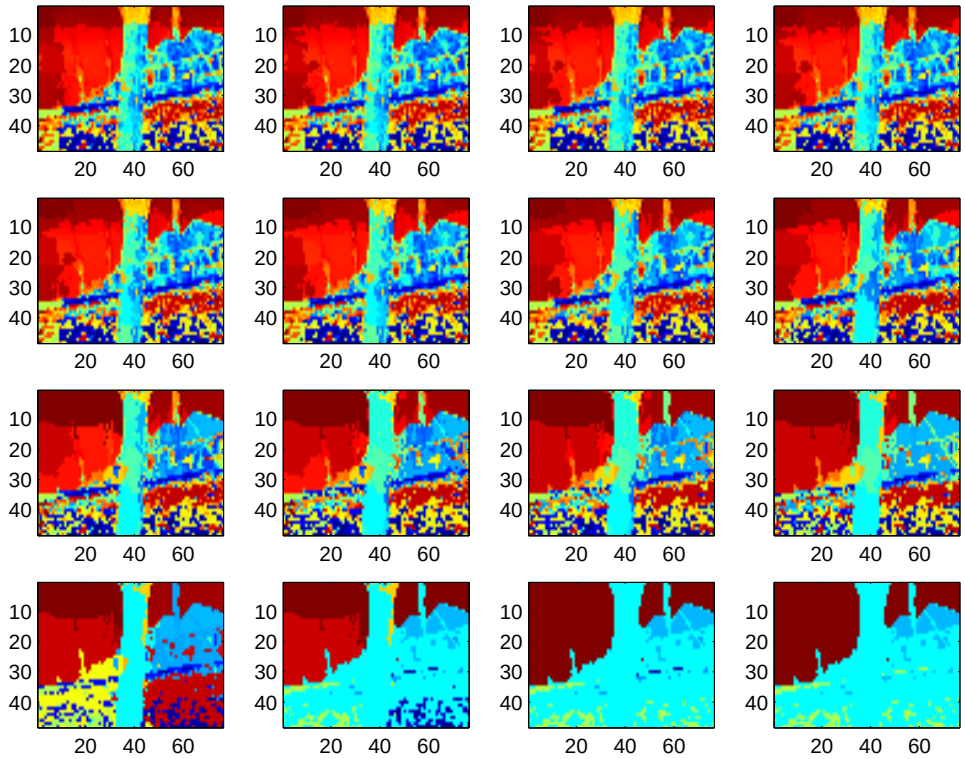


Figure 10: Evolution of the segmentation for one of the frames of the Flower Garden sequence during the hierarchical mean shift process.

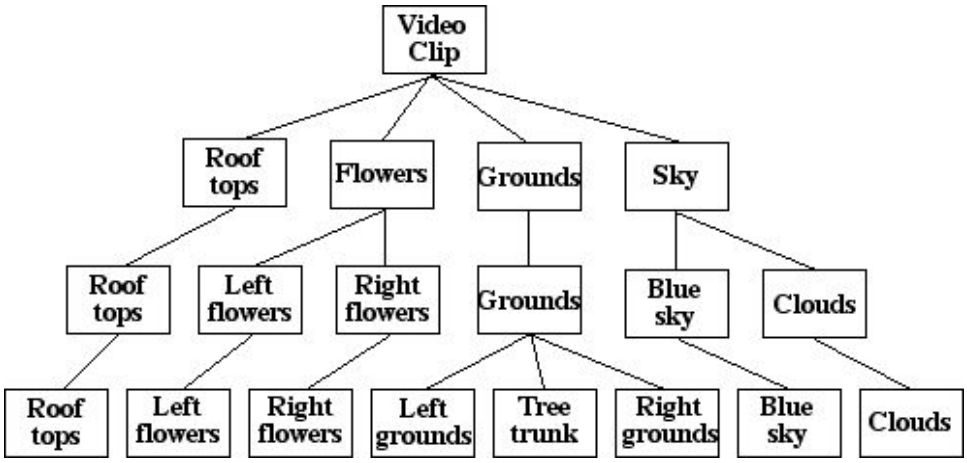


Figure 11: Tree structure describing the largest region groupings of the hierarchical mean shift analysis.

Flower Garden video clip. Regions obtained by spatio-temporal segmentation are pixel volumes in the video stack; however, for illustrative purposes we show only their cross-sections for the sixth frame of the sequence in Fig. 10. The figure shows twelve stages of the segmentation. False colors are used to represent the labels assigned to every region. To provide a labeling that better describes the merging process, a region resulting from the merging of several regions is assigned the label of the merged region that has the largest volume. Fig. 11 shows the top levels of a tree interpretation that can be constructed from the final phase of the segmentation process when the sequence is examined in the reverse of the order in which it was produced, i.e. from coarse to fine. We are investigating applications of this representation to video compression.

7 Conclusions

We have described a general algorithm for the space-time segmentation of video sequences. Our contributions to the problem of space-time segmentation are the following

1. As suggested by the Hough transform, we define pixel positions by their normal distances to motion trajectories, which are approximately invariant for pixels that belong to the same moving patch.

2. Hierarchical mean shift analysis has lower empirical complexity than standard mean shift analysis for useful radii when range search is optimized with a binary tree structure.

3. Regions are simultaneously segmented and tracked by the same mechanism; a single parameter is specified by the user.

4. Cluster centers are compact descriptors characterizing the video strands, and are useful for video indexing and retrieval.

We have several areas of further study in mind. First, the boundaries of moving regions are jagged, and arguably do not look as good as those produced by, e.g., color-texture segmentation [14]. (Note, however, that segmentation code generally includes a post-processing phase which merges small regions with larger neighbors and smooths boundaries, while we want to show how far our hierarchical mean shift approach can go all by itself.) Cleaner boundaries could be obtained if the motion components of the feature vector were given lower weight than the color components in low-confidence motion field regions. Second, in the spirit of scale-space analysis, we can analyze our hierarchical segmentation to discover which regions remain stable through increasing mean shift radii, and give preference to these regions in the segmented output, as suggested by Leung et al. [24]; then there would remain no parameter to specify. Third, we need to develop efficient ways of using video strands and hierarchical segmentation for indexing and retrieval of large video data sets, and for compression.

References

[1] E.H. Adelson and J.R. Bergen, "Spatiotemporal Energy Models for the Perception of Motion", J. Opt. Soc. Am. A., vol. 2, pp. 284–299, 1985.

[2] Z. Aghbari, K. Kaneko and A. Makinouchi, "Modeling and Querying Videos by Content Trajectories", IEEE Int. Conf. on Multimedia and Expo (ICME2000), New York, 2000.

[3] M. Allmen and C.R. Dyer, "Computing Spatiotemporal Relations for Dynamic Perceptual Organization", CVGIP: Image Understanding, vol. 58, pp. 338-351, 1993.

[4] M. Black, "Combining Intensity and Motion for Incremental Segmentation and Tracking over Long Image Sequences", ECCV, pp. 485–493, 1992.

[5] R. Bolles, H. Baker, and D. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion", Int. J. of Computer Vision, vol. 1, pp. 7–55, 1987.

[6] T. E. Boult and L. G. Brown, "Factorization-based Segmentation of Motions", Proc. IEEE Workshop on Motion Understanding, pp. 179–186, 1991.

[7] D. Comaniciu and P. Meer, "Mean Shift Analysis and Applications", Proc. IEEE Int. Conf. on Computer Vision, pp. 1197-1203, 1999.

[8] D. Comaniciu and P. Meer, "Distribution-Free Decomposition of Multivariate Data", Pattern Analysis and Applications, Vol. 2, pp. 22-30, 1999.

[9] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis", IEEE Trans. on PAMI, vol. 24, pp. 603–619, 2002.

[10] A. Del Bimbo, P. Pala and L. Tanganelli, "Video Retrieval based on Dynamics of Color Flows", ICPR, vol. 1, pp. 851–854, 2000.

[11] G. Baldi, C. Colombo, and A. Del Bimbo, "A Compact and Retrieval-Oriented Video Representation Using Mosaics", Proc. Int. Conf. on Visual Information Systems, pp. 171–178, 1999 (LNCS 1614, Springer).

[12] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering", IEEE Trans. on PAMI, vol. 17, pp. 790–799, 1995.

[13] J. Costeira and T. Kanade, "A Multi-Body Factorization Method for Motion Analysis", Proc. IEEE Intl. Conf. on Computer Vision, pp. 1071–1076, 1995.

[14] Y. Deng and B.S. Manjunath, "Unsupervised Segmentation of Color-Texture Regions in Images and Video", IEEE Trans. on PAMI, vol. 23, pp. 800–810, 2001.

[15] C. Fowlkes, S. Bellongie and J. Malik, "Efficient Spatiotemporal Grouping using the Nystrom Method", CVPR, pp. I-231–238, 2001.

[16] Fukunaga, K., "Introduction to Statistical Pattern Recognition", Academic Press, 1990.

[17] C. W. Gear, "Feature Grouping in Moving Objects", Proc. Workshop on Motion of Non-Rigid and Articulated Objects, Austin, 1994.

[18] N. Ichimura, "Motion Segmentation based on Factorization Method and Discriminant Criterion", Proc. IEEE Int. Conf. on Computer Vision, pp. 600–605, 1999.

[19] S. Gepshtein and M. Kubovy, "The Emergence of Visual Objects in Space-Time", Proc. National Academy of Sciences, USA, vol. 97, pp. 8186–8191, 2000.

[20] H. Greenspan, J. Goldberger, and A. Mayer, "A Probabilistic Framework for Spatio-Temporal Video Representation and Indexing", ECCV, pp. 461–475, 2002 (LNCS 2353, Springer).

[21] Q. Ke and T. Kanade, "A Subspace Approach to Layer Extraction", CVPR, pp. I-255–262 , 2001.

[22] N. Jojic and B. Frey, "Learning Flexible Sprites in Video Layers", CVPR, pp. I-199–206 , 2001.

[23] S. Khan and M. Shah, "Object Based Segmentation of Video Using Color, Motion and Spatial Information", CVPR, pp. II-746–751, 2001.

[24] Y. Leung, J-S. Zhang and Z-B. Xu, "Clustering by Scale-Space Filtering", IEEE Trans. on PAMI, vol. 22, pp. 1396–1410, 2000.

[25] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", Proc. DARPA Image Understanding Workshop, pp. 121–130, 1981.

[26] R. Megret and J.-M. Jolion, "Representation of Dynamic Video Content by Tracking of Grey Level Blobs", RFIA 2002.

[27] C. Merkwirth, U. Parlitz and W. Lautherborn, "Fast Nearest-Neighbor Searching for Nonlinear Signal Processing", Phys. Review E., vol. 62, pp. 2089–2097, 2000. TSTool package available at http://www.physik3.gwdg.de/tstool/

[28] F. Moscheni, S. Bhattacharjee and M. Kunt, "Spatiotemporal Segmentation Based on Region Merging", IEEE Trans. on PAMI, vol. 20, pp. 897–915, 1998.

[29] H. S. Sawhney and S. Ayer, "Compact Representations of Videos through Dominant and Multiple Motion Estimation", IEEE Trans. on PAMI, vol. 18, pp. 814–830, 1996.

[30] J. Shi and J. Malik, "Motion Segmentation and Tracking using Normalized Cuts", IJCV, pp. 1154–1160, 1998.

[31] T. Syeda-Mahmood, A. Vasilescu and S. Sethi, "Recognizing Action Events from Multiple Viewpoints", Proc. IEEE Workshop on Detection and Recognition of Events in Video, pp. 64–73, 2001.

[32] W.B. Thompson, "Combining Motion and Contrast for Segmentation", IEEE Trans. on PAMI, pp. 543–549, 1980.

[33] P. H. S. Torr and A. Zisserman, "Concerning Bayesian Motion Segmentation, Model Averaging, Matching and the Trifocal Tensor", ECCV, pp. I-511–527, 1998.

[34] J. Y. A. Wang and E. A. Adelson, "Representing Moving Images with Layers", IEEE Trans. on Image Processing, vol. 3, pp. 625-638, 1994.

[35] D. Zhong and S-F. Chang, "Spatio-Temporal Video Search Using the Object Based Video Representation", ICIP, pp. 21–24, 1997.